BookML: a bookdown flavoured GitBook port for LaTeXML

Vincenzo Mantova

23 August 2021

Abstract

This package is mainly a port of the GitBook style of bookdown for LaTeXML, but it also offers a few extra features and bugfixes for LaTeXML that can be used with or without the GitBook style:

- transparent generation of SVG pictures via LaTeX for packages not well supported by LaTeXML, such as TikZ pictures, animations, XY-matrices;
- a simple method to add alternative text for images;
- partial support for arbitrary HTML content;
- backport of some minor styling fixes from LaTeXML 0.8.6 such as mobile friendliness as well as the occasional bookdown bugfixes;
- direct embedding of MathJax, with the option of choosing between versions 2 and 3 or disabling it.

Formats: GitBook (html), plain with Computer Modern (html), PDF.

Contents

1 Getting started

1. Install the prerequisites:
   - **Required:** a working LaTeXML installation, at least version 0.8.5.
   - **Optional** (necessary for generating the images via LaTeX): a working TeX install containing `dvisvgm` and `latexmk`.
2. Unpack the latest BookML release in the same folder as your `.tex` files (and not in a subfolder).
3. Anywhere between `\documentclass` and `\begin{document}`, add `\usepackage{bookml}`.
4. Compile your file to HTML:

   ```
   latexmlc --navigationtoc=context \
     --dest=my_latex_file/index.html \
     my_latex_file.tex
   ```

   Add `--splitat=section` (or chapter, part...) to split the content on multiple pages. See the LaTeXML manual for all the options.

2 Options

The `bookml` package accepts a few options (for instance use `\usepackage[style=plain,nomathjax]{bookml}` to disable the GitBook style and to avoid including MathJax). The options have no effect on the PDF output.

You can also pass these options at compilation time using the `--preload` option of `latexml` and `latexmlc`:

```
latexmlc --preload=[style=plain,nomathjax]bookml \
  --dest=my_latex_file/index.html \
  my_latex_file.xml
```

`style=gitbook` Use the GitBook style (the default behaviour). When using the GitBook style, you must call `latexmlc` (or `latexmlpost`) with the option `--navigationtoc=context`.

`style=plain` Use the LaTeXML style with a few slightly opinionated tweaks.

`style=none` Use the LaTeXML style with no tweaks (except for backported styles and some fixes).

`nomathjax` Do not include MathJax in the output.

`mathjax=2` Use MathJax version 2 instead of version 3.

`imagescale=X.XXX` Rescale the images generated via LaTeX (§ 4.4) by the desired factor. The scaling factor is adjusted internally based on the options `8pt`, `9pt`, ..., `12pt` being passed to the document class.

`download=docs.pdf|docs.epub` |-separated list of files to list under bookdown's download button (only valid for the GitBook style). The default value is `\jobname.pdf`, where `\jobname` is the name of the `.tex` file being compiled without the extension. Use `download=` to remove the download button. For now, only PDF and EPUB files are supported.

3 Customisation

3.1 CSS and fonts

Just create a `bmluser` folder and add any `.css` file to it. The files will be included at the end of the `<head>` tag and override the previous styles.

If the name of a file contains the substring `.gitbook.`, then that file will be used only when the `style=gitbook` option is passed (likewise for `.plain.` or `.none.`).

The plain version of this document has been compiled with an additional `computer-modern.plain.css` that sets the font to Computer Modern.

3.2 HTML output

You can modify `LaTeXML-html5.xsl` to change the HTML output. Just be careful not to overwrite the file during updates.

4 Commands

4.1 Conditional execution

Call `\iflatexml ... \else ... \fi` to write code that is executed only by LaTeXML, or only (pdf)LaTeX respectively. `bookml` will try to use the `latexml` package, which offers the same functionality (and much more), if available. See Figure 1.

```
\caption{Example of
  \iflatexml\ltxinline|\xymatrix|\else\ltxinline|\\xymatrix|\fi{}
  from the \ltxinline|xypic| documentation.}
```

Figure 1: Example of `\iflatexml`, used in Figure 3 to work around a subtle difference between LaTeXML and LaTeX.

4.2 Alternative text for images

Call `\bmlDescription{textual description}` *right after* an image to populate its `alt` attribute (or `aria-label` if appropriate). Inspect the HTML source of Figure 2 or use a screen reader to check its text description.

4.3 Add custom CSS classes

Call `\bmlPlusClass{class}` *right after* some piece of content to add a CSS class. If done within text, its effect may be unpredictable. Its main use is to call `\bmlPlusClass{bml_no_invert}` after an image to prevent the picture from getting inverted in night mode. Compare how Figure 2 (with `bml_no_invert`) and Figure 3 (no additional classes) change in night mode to see the difference.

Note that the package `latexml` also offers `\lxAddClass` and `\lxWithClass` for the same effect but different behaviour regarding which element gets the class.

```
\begin{animateinline}[
  alttext=none,loop,controls,poster=20,autopause,
  begin={\begin{tikzpicture}
    \useasboundingbox (0,0) rectangle (4,3);},
    end={\end{tikzpicture}}]{30}
  \multiframe{160}{dShift=40mm+-0.5mm}
    {\duck[tophat,xshift=\dShift]}
\end{animateinline}
\bmlDescription{A stylised rubber duck, yellow
  and wearing a black top hat, enters from the right
  and slides until it exits from the left. The animation
  repeats every six seconds.}
\bmlPlusClass{bml_no_invert} % preserve colours in night mode
```

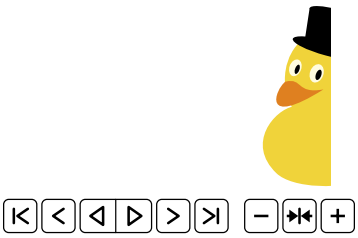

Figure 2: A fancy duck. Click on the image to start the animation (for the PDF, it requires a compatible software such as Acrobat Reader).

4.4 Generate pictures with LaTeX

LaTeXML supports the `picture` environment as well as *some* Ti*k*Z pictures, but not all which will come out mangled, and some common packages are not supported altogether (for instance `xypic`, `tikzcd`, `animate`).

BookML offers a simple automated way of generating SVG images using LaTeX, bypassing LaTeXML entirely. In your preamble, after `\usepackage{bookml}`, write

```
\bmlImageEnvironment{tikzpicture}
\bmlImageEnvironment{animateinline}

% optional, but strongly recommended:
% do not load tikz when running in LaTeXML
\iflatexml\else
\usepackage{tikz}
\usepackage{animate}
\fi
```

Now every `tikzpicture` and `animate` environment will be compiled with LaTeX (using `latexmk`) and converted to SVG images (using `dvisvgm`). Figure 2 demonstrates this approach.

If you only need this mechanism in a pinch, you can simply wrap the desired content between `\begin{bmlimage}` and `\end{bmlimage}` as exemplified in Figure 3.

```
\begin{bmlimage}
  \[ \xymatrix{
       U \ar@/_/[ddr]_y \ar@/^/[drr]^x \ar@{.>}[dr]|-{(x,y)} \\
       & X \times_Z Y \ar[d]^q \ar[r]_p & X \ar[d]_f \\
       & Y \ar[r]^g & Z} \]
\end{bmlimage}
```

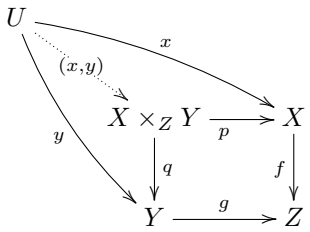

Figure 3: Example of \xymatrix from the xypic documentation.

4.5 Direct HTML input

You can insert arbitrary HTML code using \bmlRawHTML{html code}.

Warning: the HTML code needs to be written in 'XML syntax', so you have to close all the tags (for instance, write
 instead of
, close the <p> tags, and so on) and empty attributes *must* be given the value "" (see this old W3C guide for some indications). Moreover, you must remember to escape your %&_^${}, and replace \ with \textbackslash.

\bmlRawHTML is robust, i.e. it does not change the category codes, so it can be used inside \newcommand to create custom macros. See for instance Figure 4 for a generic YouTube embedding macro. Note that the video will not be visible in the PDF, so a link should always be provided (possibly PDF only, as in the example).

4.6 Wrapping LaTeX in HTML tags (experimental)

The command \bmlHTMLEnvironment{tag} defines an environment \begin{h:tag} ... \end{h:tag} which wraps the content between <tag>...</tag>. One can also add attributes as optional arguments \begin{h:tag}[attr1=val1,attr2=val2].

This approach is very fragile because LaTeXML cannot cope well with unknown tags, and the implementation may change in the future. For now, this is best used for tags which behave like blocks and can contain paragraphs. See Figure 5 for an example.

```
\newcommand{\youtube}[2]{\bmlRawHTML{
  <div style="max-width: 1920px; width: 100\%">
    <div style="position: relative;
        padding-bottom: 56.25\%; height: 0; overflow: hidden;">
      <iframe width="1920" height="1080"
        src="https://www.youtube-nocookie.com/embed/#1"
        title="YouTube: #2"
        frameborder="0" allowfullscreen=""
        style="border:none; position: absolute; top: 0; left: 0;
          right: 0; bottom: 0; height: 100\%; max-width: 100\%;"
        allow="accelerometer; autoplay; clipboard-write;
          encrypted-media; gyroscope; picture-in-picture"/>
    </div>
  </div>}
\iflatexml\else
\begin{center}
  Watch \href{https://www.youtube.com/watch?v=#1}{#2}.
\end{center}
\fi}
\youtube{mH0oCDa74tE}
  {Group theory, abstraction, and the 196,883-dimensional monster}
```

Watch Group theory, abstraction, and the 196,883-dimensional monster.

Figure 4: Demonstration of `\bmlRawHTML` within `\newcommand` with a video from 3Blue1Brown.

Code for `<details>`.
Completing the quine is left as an exercise for the reader.

```
\bmlHTMLEnvironment{details}
\begin{h:details}[style={text-align: left; width: 100\%}]
  % <summary> cannot contain <p>, better use \bmlRawHTML
  \bmlRawHTML{<summary>Code for
    <code>\&lt;details\&gt;</code>.</summary>}
  \iflatexml\else {\bfseries Code for
    \lstinline[language=html,frame=none]|<details>|.}\fi

  Completing the quine is left as an exercise for the reader.
\end{h:details}
```

Figure 5: Implementation of the `<details>` tag.